Di-Cones

Parker Emmerson

November 2024

1 Introduction

We have thoroughly investigated folding a circle into a single cone. However, the next, logical step is to use the, "angle removed," to form a second cone. Instead of wasting the arc length, this arc length can be used. So, essentially, we have a way of deleting space and expanding space, as the two cone system will by implication expand space when each individual cone is expanded into its respective unique, "origin," circle, we will essentially delete space in front of a craft and expand that same space behind the craft. Obviously, this is theoretical, however, the mathematics is very real. Additionally, it should be possible to generate n-cones from a single circle, though to parameterize them such that they intersect at a single point on their bases, share a single line along their slant heights, and a single exact apex point remains elusive. This 2-cone system serves as a starting point.

2 Code

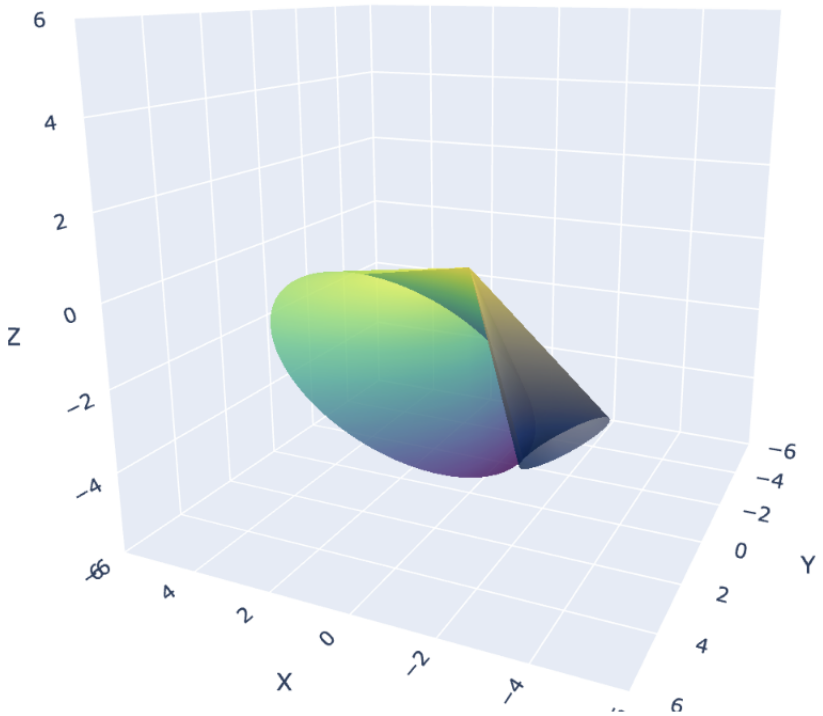
```

# Install necessary packages
!pip install plotly ipywidgets

# Enable custom widget manager (if in Google Colab)
try:
    from google.colab import output
    output.enable_custom_widget_manager()
except ImportError:
    pass

import numpy as np
import plotly.graph_objects as go
import ipywidgets as widgets
from ipywidgets import interact

def generate_cone(base_radius, height, rotation_matrix, num_points=100):
    theta = np.linspace(0, 2 * np.pi, num_points)
    z = np.linspace(0, height, num_points)
    theta_grid, z_grid = np.meshgrid(theta, z)

    r_grid = base_radius * (z_grid / height)
    x_grid = r_grid * np.cos(theta_grid)
    y_grid = r_grid * np.sin(theta_grid)
    z_grid = -z_grid  # The cone's apex is at the origin, opening upward

    # Apply rotation using the rotation matrix
```

```
    x_rot = rotation_matrix[0, 0] * x_grid + rotation_matrix[0, 2] *
        z_grid
    y_rot = y_grid  # unchanged as we rotate around y-axis
    z_rot = rotation_matrix[2, 0] * x_grid + rotation_matrix[2, 2] *
        z_grid

    return x_rot, y_rot, z_rot

def rotation_matrix_y(angle):
    """Return rotation matrix for rotation around the Y-axis."""
    return np.array([
        [np.cos(angle), 0, np.sin(angle)],
        [0, 1, 0],
        [-np.sin(angle), 0, np.cos(angle)]
    ])

def visualize_cones_with_single_intersection(r, theta):
    # Calculate the radii and heights of the two cones
    r1 = r - (r * theta) / (2 * np.pi)
    eta1 = np.sqrt(r**2 - r1**2)

    r2 = (r * theta) / (2 * np.pi)
    eta2 = np.sqrt(r**2 - r2**2)

    # Calculate the angle for a single intersection
    shared_angle = np.arctan(r1 / eta1) + np.arctan(r2 / eta2)

    # Create rotation matrices to align the cones
    rotation_matrix1 = rotation_matrix_y(-shared_angle / 2)
    rotation_matrix2 = rotation_matrix_y(shared_angle / 2)

    # Generate cone geometries
    X1, Y1, Z1 = generate_cone(r1, eta1, rotation_matrix1)
    X2, Y2, Z2 = generate_cone(r2, eta2, rotation_matrix2)

    # Plotting
    surface1 = go.Surface(x=X1, y=Y1, z=Z1, colorscale='Viridis', opacity
        =0.8, showscale=False)
    surface2 = go.Surface(x=X2, y=Y2, z=Z2, colorscale='Cividis', opacity
        =0.8, showscale=False)

    layout = go.Layout(
        title='Right Cones with Circular Bases Sharing an Apex',
        scene=dict(
            xaxis=dict(title='X', range=[-6, 6]),
            yaxis=dict(title='Y', range=[-6, 6]),
            zaxis=dict(title='Z', range=[-6, 6]),
            aspectmode='cube',  # This ensures that the box is a true cube
        ),
        width=700,
        height=700,
    )

    fig = go.Figure(data=[surface1, surface2], layout=layout)
```

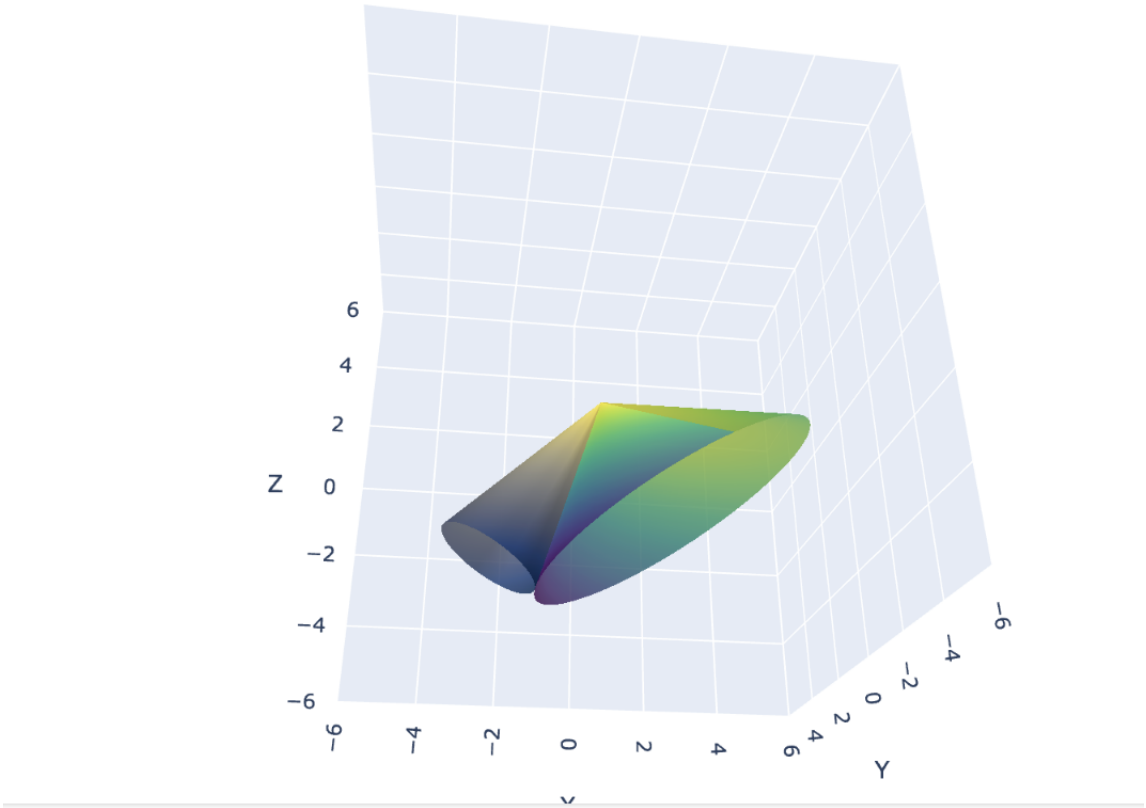
```
    fig.show()

# Interactive sliders setup
theta_slider = widgets.FloatSlider(
    value=np.pi / 2,
    min=0.1,
    max=2 * np.pi - 0.01,
    step=np.pi / 180,
    description='Theta (rad):',
    continuous_update=True,
    readout_format='.2f',
)

radius_slider = widgets.FloatSlider(
    value=5.0,
    min=1.0,
    max=10.0,
    step=0.1,
    description='Radius (r):',
    continuous_update=True,
    readout_format='.1f',
)

# Interactive visualization
interact(visualize_cones_with_single_intersection, r=radius_slider, theta=
    theta_slider);
```

3 Visualizations

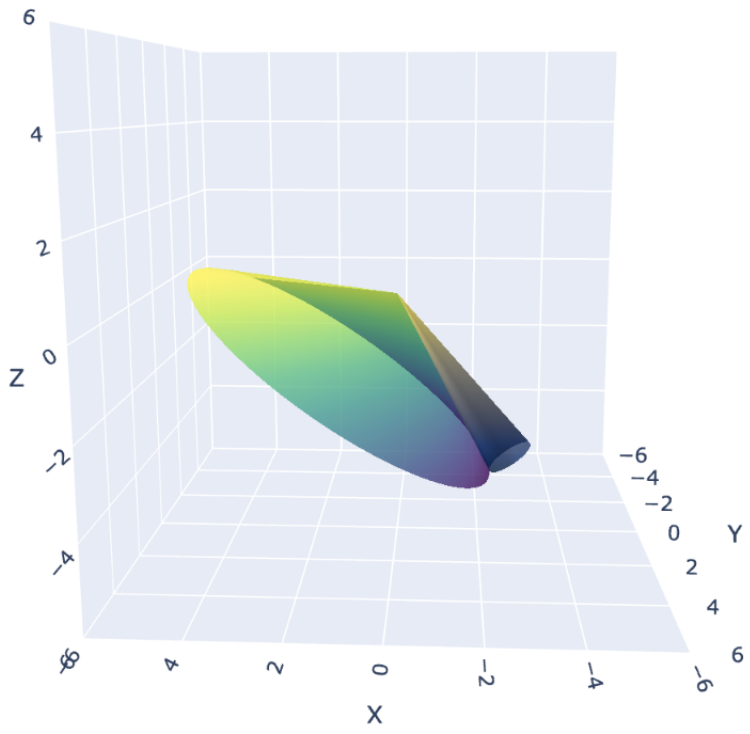
Right Cones with Circular Bases Sharing an Apex



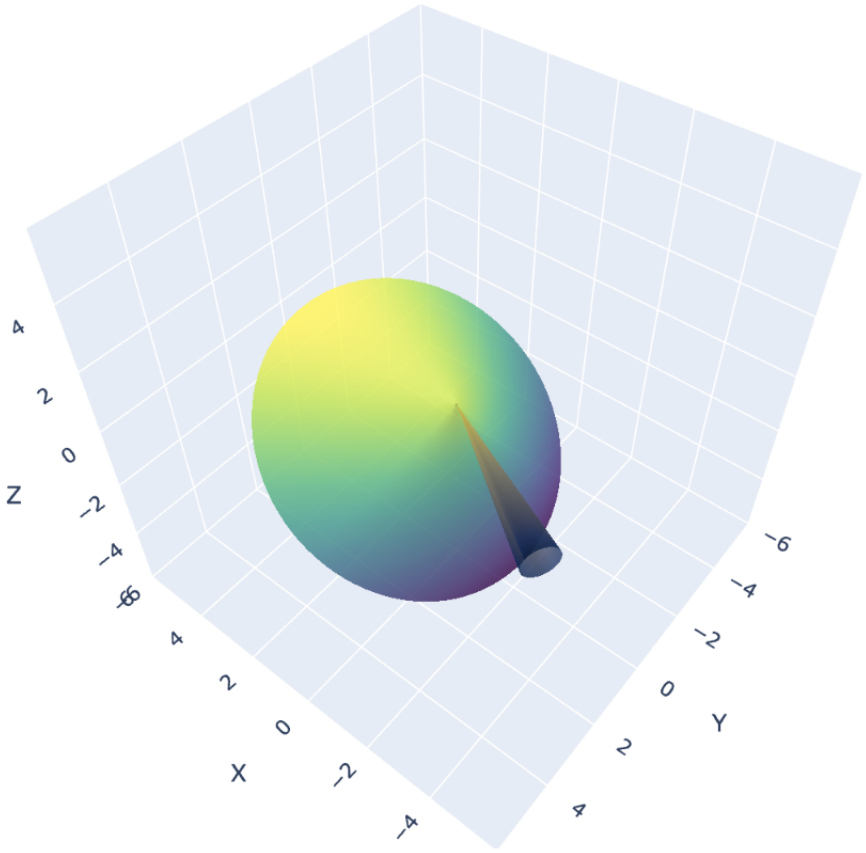
Right Cones with Circular Bases Sharing an Apex



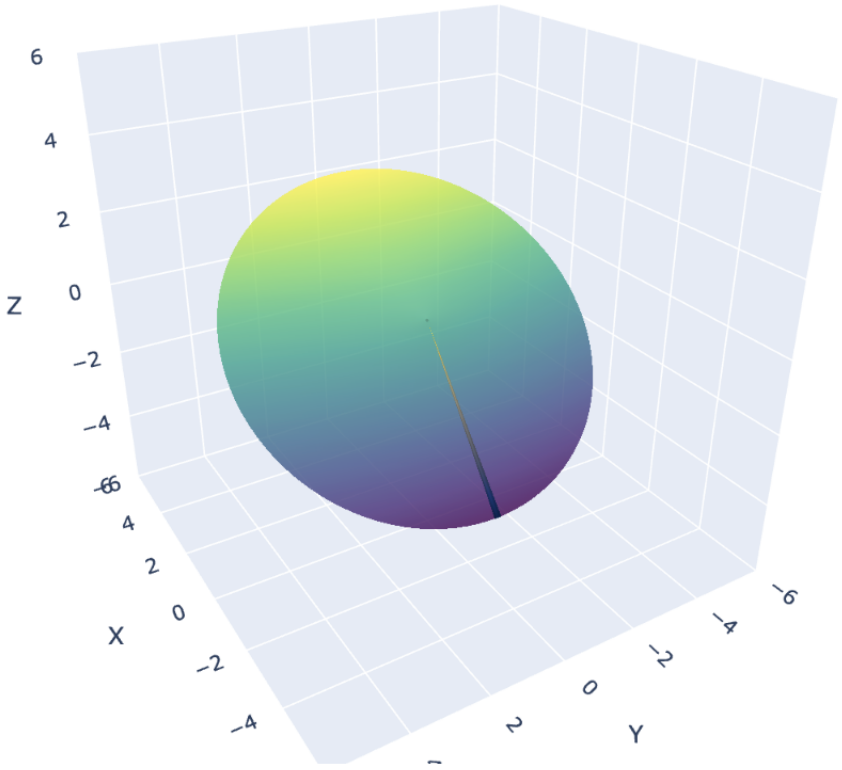
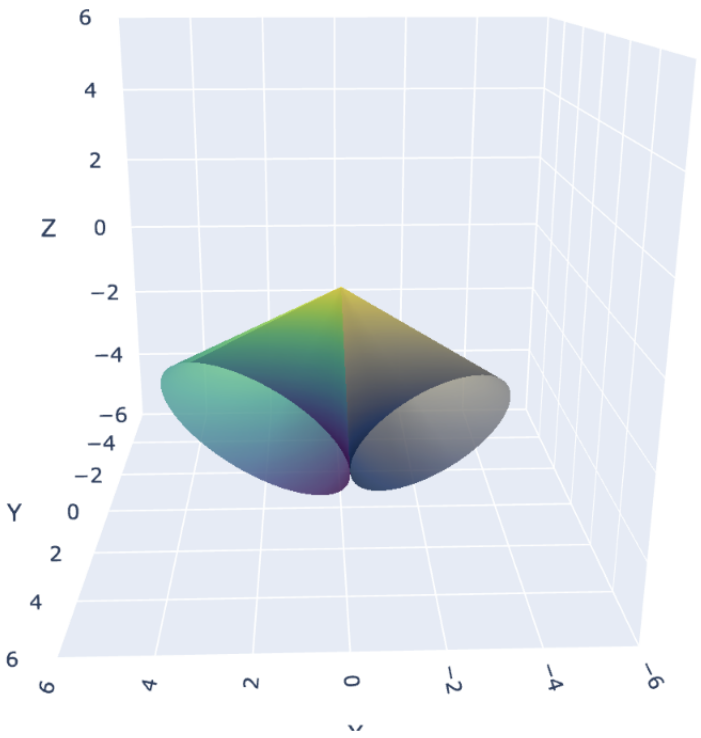
Right Cones with Circular Bases Sharing an Apex



Right Cones with Circular Bases Sharing an Apex



Right Cones with Circular Bases Sharing an Apex



Right Cones with Circular Bases Sharing an Apex